EXPTIME-complete Decision Problems for Modal and Mixed Specications

Antonik, Adam; Huth, Michael; Larsen, Kim Guldstrand; Nyman, Ulrik; Wasowski, Andrzej

Published in:
Electronical Notes in Theoretical Computer Science

DOI (link to publication from Publisher):
10.1016/j.entcs.2009.06.011

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

EXPTIME-complete Decision Problems for Mixed and Modal Specifications*

Adam Antonik[1], Michael Huth[1],
Kim G. Larsen[2], Ulrik Nyman[2], and Andrzej Wąsowski[3,2]

[1] Department of Computing, Imperial College London, United Kingdom
{aa1001,mrh}@doc.imperial.ac.uk
[2] Department of Computer Science, Aalborg University, Denmark
{kgl,ulrik}@cs.aau.dk
[3] IT University of Copenhagen, Denmark
wasowski@itu.dk

Abstract. Mixed and modal transition systems are formalisms allowing mixing of over- and under-approximation in a single specification. We show EXPTIME-completeness of three fundamental decision problems for such specifications: whether a set of mixed or modal specifications has a common implementation, whether a sole mixed specification has an implementation, and whether all implementations of one mixed specification are implementations of another mixed or modal one. These results are obtained by a chain of reductions starting with the acceptance problem for linearly bounded alternating Turing machines.

1 Introduction

Behavioral models capture actual, desired or required system behavior and can so serve as documentation, specification or as the basis of analysis and validation activities. Formal behavioral models — of which we mention process algebras, Petri nets and labelled transition systems — bring a high degree of rigor and dependability to validation and verification activities.

Often one has to deal with more than one behavioral model at a time. For example, in requirement elaboration one may have several versions of a model, in component-based design one may have models that each focus on a different aspect of the system, and in formal verification one may have a system model accompanied by models that represent either desired features or genuinely faulty behavior. In each of these cases the modeller may want to have assurance that this collection of models is consistent. If versions of models are inconsistent with each other, this may reveal important implementation trade-offs. If all aspect models are inconsistent, their combination is not implementable. If a system model is inconsistent with all members of a given set of fault models, the system will not exhibit any of these flaws. Finally if a system model is consistent with

* Partially supported by the UK EPSRC projects EP/D50595X/1 and EP/E028985/1

a set of feature models, then the system will be able to actually implement all these features.

A related concept is the consistency of a *single* behavioral model. If models serve as specifications, their inconsistency suggests that the specification cannot be implemented. Conversely, a consistent model boosts our confidence in implementability and may even allow code-generation of such an implementation.

The stepwise-refinement paradigm proposes to write specifications as models and to then repeatedly refine such models until an implementation has been realized. In a *thorough* interpretation, refinement is decreasing the set of possible implementations: only implementations that were possible before the refinement step are still possible thereafter, but not necessarily all of them anymore.

This paper is devoted to studying the exact computational complexity of these three decision problems; whether finitely many models are consistent, whether a single model is consistent, and whether one model thoroughly refines another. The actual models we study are *mixed specifications* — stateful models with allowed and required transitions, well recognized as a formal foundation for system specification and abstraction alike [1–10]. We show that

- deciding whether finitely many modal or mixed specifications are consistent is EXPTIME-complete in the sum of the sizes of these specifications
- deciding whether one mixed specification is consistent is EXPTIME-complete in the size of that specification
- deciding whether one mixed specification thoroughly refines another mixed specification is EXPTIME-complete in the sum of their sizes.

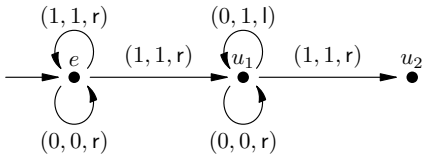
Interestingly, checking the consistency of 100 mixed specifications with a few states each can be dramatically more complex than checking the consistency of a few mixed specifications with 100 states each. This is in striking contrast to the situation when all mixed specifications are fully refined (have identical required and allowed behaviors). In that case, consistency checks reduce to pairwise bisimilarity checks, which can be performed in polynomial time.

Our complexity results motivate future research that aims to either approximate these three decision problems soundly and efficiently, or that identifies sub-classes of specifications for which these decision problems are less complex.

We proceed by introducing the necessary background on alternating Turing machines, specifications, and their decision problems in Section 2. In Section 3 state-of-the-art bounds for these problems are reported. The new EXPTIME-completeness results are given in Section 4. Section 5 reflects on a remaining open complexity gap for a special kind of mixed specifications, modal ones. We conclude in Section 6.

Related work We refer to our recent overview [11] for a full account of related work. The present paper primarily improves on the results of [12], which are discussed in detail in Section 3. The relation of this work to generalized model checking [13] is detailed in Section 5.

In [14] a superpolynomial algorithm is given, which establishes common implementation for $k > 1$ modal specifications. The algorithm is exponential in k,



$$\delta(e,\ 0) = \{(e,\ 0, \mathsf{r})\}$$
$$\delta(e,\ 1) = \{(e,\ 1, \mathsf{r}), (u_1, 1, \mathsf{r})\}$$
$$\delta(u_1, 0) = \{(u_1, 1, \mathsf{l}), (u_1, 0, \mathsf{r})\}$$
$$\delta(u_1, 1) = \{(u_2, 1, \mathsf{r})\}$$
$$\delta(u_2, 0) = \delta(u_2, 1) = \{\}$$

Fig. 1. The transition relation of an ATM as a labelled graph and a function.

but polynomial if k is fixed. It computes a common implementation if one exists. These upper bounds follow also from the polynomial algorithm for consistency checking of a conjunction of disjunctive modal transition systems, as studied in [3].

In [15] Hussain and Huth present an example of two modal specifications that have a common implementation but no greatest common implementation.

Fischbein et al. [16] use modal specifications for behavioral conformance checking of products against specifications of product families. They propose a new thorough refinement whose implementations are defined through a generalization of branching bisimulation. The thorough refinement obtained in this manner is finer than weak refinement, and argued to be more suitable for conformance checking.

2 Background

Let us begin with a definition of the decision problem used in the main proof of this paper. An *Alternating Turing Machine* [17], or an ATM, is a tuple $T = (Q, \Gamma, \delta, q_0, \mathsf{mode})$, where Q is a non-empty finite set of control states, Γ is an alphabet of tape symbols, $\mathsf{null} \notin \Gamma$ is a special symbol denoting empty cell contents, $\delta\colon Q \times (\Gamma \cup \{\mathsf{null}\}) \to \mathcal{P}(Q \times \Gamma \times \{\mathsf{l}, \mathsf{r}\})$ is a transition relation, $q_0 \in Q$ is the initial control state, and $\mathsf{mode}\colon Q \to \{\mathsf{Univ}, \mathsf{Exst}\}$ is a labeling of control states as respectively universal or existential. Universal and existential states with no successors are called accepting and rejecting states (respectively). Each ATM T has an infinite tape of cells with a leftmost cell. Each cell can store one symbol from Γ. A head points to a single cell at a time, which can then be read or written to. The head can then move to the left or right: $(q', a', \mathsf{r}) \in \delta(q, a)$, e.g., says "if the head cell (say c) reads a at control state q, then a successor state can be q', in which case cell c now contains a' and the head is moved to the cell on the right of c." The state of the tape is an infinite word over $\Gamma \cup \{\mathsf{null}\}$.

Figure 1 presents an example of an ATM T over a binary alphabet $\Gamma = \{0, 1\}$ where arrows $q \xrightarrow{(a,a',d)} q'$ denote $(q', a', d) \in \delta(q, a)$. The initial control state e is an existential one, and both u_i control states are universal.

Configurations of an ATM T are triples $\langle q, i, \tau \rangle$ where $q \in Q$ is the current control state, the head is on the ith cell from the left, and $\tau \in (\Gamma \cup \mathsf{null})^\omega$ is the current tape state. For input $w \in \Gamma^*$, the initial configuration is $\langle q_0, 1, w\mathsf{null}^\omega \rangle$.

The recursive and parallel execution of all applicable[4] transitions δ from initial configuration $\langle q_0, 1, w\mathsf{null}^\omega\rangle$ yields a computation tree $\mathsf{T}_{\langle T,w\rangle}$. We say that ATM T accepts input w iff the tree $\mathsf{T}_{\langle T,w\rangle}$ accepts, where the latter is a recursive definition:

- $\mathsf{T}_{\langle T,w\rangle}$ with root $\langle q, i, \tau\rangle$ and $\mathsf{mode}(q) = \mathsf{Exst}$ accepts iff there is a successor $\langle q', i', \tau'\rangle$ of $\langle q, i, \tau\rangle$ in $\mathsf{T}_{\langle T,w\rangle}$ such that the sub-tree with root $\langle q', i', \tau'\rangle$ accepts
- $\mathsf{T}_{\langle T,w\rangle}$ with root $\langle q, i, \tau\rangle$ and $\mathsf{mode}(q) = \mathsf{Univ}$ accepts iff for all successors $\langle q', i', \tau'\rangle$ of $\langle q, i, \tau\rangle$ in $\mathsf{T}_{\langle T,w\rangle}$ the sub-tree with root $\langle q', i', \tau'\rangle$ accepts (in particular, this is the case if there are no such successors)

The ATM of Figure 1 accepts the regular language $(0 + 1)^*10^*1(0 + 1)^*$. Observe that u_2 is the only accepting state. Intuitively the part of T rooted in e accepts the prefix $(0 + 1)^*1$ — the semantics of existential states is locally that of states in non-deterministic Turing machines. The part of T rooted in u_1 consumes a series of 0 symbols until 1 is reached, which leads to acceptance. The suffix of the input word after the last 1 is ignored. Note that the computation forks in u_1 whenever a 0 is seen. However, the top branch would reach the earlier 1 eventually and accept.

An ATM T is *linearly bounded* iff for all words $w \in \Gamma^*$ accepted by T, the accepting part of the computation tree $\mathsf{T}_{\langle T,w\rangle}$ only contains configurations $\langle q, i, v\mathsf{null}^\omega\rangle$, where the length of $v \in \Gamma^*$ is no greater than the length of w. That is to say, by choosing exactly one accepting successor for each existential configuration in $\mathsf{T}_{\langle T,w\rangle}$, and by removing all the remaining successors and configurations unreachable from the root, one can create a smaller tree that only contains configurations with $\langle q, i, v\mathsf{null}^\omega\rangle$ where $|v| \leq |w|$. We refer to such pruned computation trees simply as "computations".

Our notion of "linear boundedness" follows [18] in limiting the tape size to the size of the input. This limitation does not change the hardness of the acceptance problem (see below). In addition we assume that linearly bounded ATMs have no infinite computations since any linearly bounded ATM can be transformed into another linearly bounded ATM, which accepts the same language, but also counts the number of computation steps used, rejecting any computation whose number of steps exceeds the number of possible configurations.[5]

Let $\mathrm{ATM}_{\mathrm{LB}} = \{\langle T, w\rangle \mid w \in \Gamma^*$ accepted by linearly bounded ATM $T\}$. The problem of deciding if for an arbitrary linearly bounded ATM T and an input w the pair $\langle T, w\rangle$ is in $\mathrm{ATM}_{\mathrm{LB}}$ is EXPTIME-complete [17].

Let us now define the basic models of interest in our study [2, 7, 20]:

Definition 1. *For a finite alphabet of actions Σ, a* mixed specification *M is a triple $(S, R^\Box, R^\Diamond)$, where S is a finite set of states and $R^\Box, R^\Diamond \subseteq S \times \Sigma \times S$ are must- and may- transitions relations (respectively). A* modal *specification is*

[4] Transitions $(\_, \_, \_, \_, \mathsf{l})$ are not applicable in configurations $\langle \_, 1, \_\rangle$ as the head cannot move over the left boundary of the tape.

[5] This is possible because ASPACE = EXPTIME [19, Thm. 10.18].

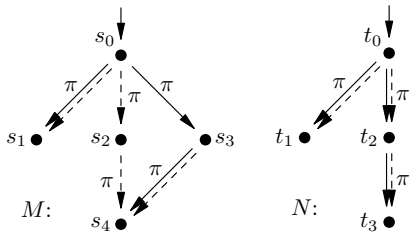


Fig. 2. Mixed $((M, s_0))$ and modal $((N, t_0))$ specifications with $I(M, s_0) = I(N, t_0)$ but not $(N, t_0) \prec (M, s_0)$.

a mixed specification satisfying $R^{\Box} \subseteq R^{\Diamond}$; all its must-transitions are also may-transitions. A pointed *mixed (respectively modal) specification (M, s) is a mixed (modal) specification M with a designated initial state $s \in S$. The size $|M|$ of a mixed (modal) specification M is defined as $|S| + |R^{\Box} \cup R^{\Diamond}|$.*

Refinement [2, 7, 20], called "modal refinement" in [9], is a co-inductive relationship between two mixed specifications that verifies that one such specification is more abstract than the other. This generalizes the co-inductive notion of bisimulation [21] to mixed specifications:

Definition 2. *A mixed specification $(N, t_0) = ((S_N, R^{\Box}_N, R^{\Diamond}_N), t_0)$* refines *another mixed specification $(M, s_0) = ((S_M, R^{\Box}_M, R^{\Diamond}_M), s_0)$ over the same alphabet Σ, written $(M, s_0) \prec (N, t_0)$, iff there is a relation $Q \subseteq S_M \times S_N$ containing (s_0, t_0) and whenever $(s, t) \in Q$ then*

1. *for all $(s, a, s') \in R^{\Box}_M$ there exists some $(t, a, t') \in R^{\Box}_N$ with $(s', t') \in Q$*
2. *for all $(t, a, t') \in R^{\Diamond}_N$ there exists some $(s, a, s') \in R^{\Diamond}_M$ with $(s', t') \in Q$*

Deciding whether one finite-state mixed specification refines another one is in P. For mixed specification (N, t_0) and modal specification (M, s_0) in Figure 2 we have $(M, s_0) \prec (N, t_0)$, given by $Q = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_2), (s_4, t_3)\}$. Note that throughout figures, solid arrows denote $R^{\Box}$-transitions, and dashed arrows denote $R^{\Diamond}$-transitions. But we do not have $(N, t_0) \prec (M, s_0)$. To see this, assume that there is a relation Q with $(t_0, s_0) \in Q$ satisfying the properties in Definition 2. Then from $(s_0, \pi, s_2) \in R^{\Diamond}_M$ we infer that there must be some x with $(t_0, \pi, x) \in R^{\Diamond}_N$ and $(x, s_2) \in Q$. In particular, x can only be t_1 or t_2. If x is t_1, then since $(s_2, \pi, s_4) \in R^{\Diamond}_M$ and $(t_1, s_2) \in Q$ there has to be some $R^{\Diamond}_N$ transition out of t_1, which is not the case. If x is t_2, then $(t_2, \pi, t_3) \in R^{\Box}_N$ and $(t_2, s_2) \in Q$ imply that there is some $R^{\Box}_M$ transition out of s_2, which is not the case. In conclusion, there cannot be such a Q and so $(N, t_0) \not\prec (M, s_0)$.

Labeled transition systems over an alphabet Σ are pairs (S, R) where S is a non-empty set of states and $R \subseteq S \times \Sigma \times S$ is a transition relation. We identify labelled transition systems (S, R) with modal specifications (S, R, R). The set of implementations $I(M, s)$ of a mixed specification (M, s) are all pointed labelled

transition systems (T, t) refining (M, s). Note that $I(M, s)$ may be empty in general, but is guaranteed to be non-empty if M is a modal specification.

Definition 3. *Let (N, t) and (M, s) be pointed mixed specifications. As in [9] we define* thorough refinement *$(M, s) \prec_{th} (N, t)$ to be the predicate $I(N, t) \subseteq I(M, s)$.*

Refinement approximates this notion: $(M, s) \prec (N, t)$ implies $(M, s) \prec_{th} (N, t)$ since refinement is transitive. The converse is known to be false [22–24]; Figure 2 provides a counterexample.

We shall now formally define the decision problems informally stated above:
Common implementation (CI): given $k > 1$ modal or mixed specifications (M_i, s_i), is the set $\bigcap_{i=1}^{k} I(M_i, s_i)$ non-empty?
Consistency (C): Is $I(M, s)$ non-empty for a modal or mixed specification (M, s)?
Thorough refinement (TR): Does a mixed specification (N, t) thoroughly refine a mixed specification (M, s), i.e., do we have $I(N, t) \subseteq I(M, s)$?

As far as these decision problems are concerned, the restriction to finite implementations, which follows from restricting our definitions to finite specifications, causes no loss of generality, as already explained in [12]. A mixed specification (M, s) is consistent in the infinite sense iff its characteristic modal mu-calculus formula $\Psi_{(M,s)}$ [25] is satisfiable. Appealing to the small model theorem for mu-calculus, $\Psi_{(M,s)}$ is satisfiable iff it is satisfiable over finite-state implementations. We can reason in a similar manner about common implementation, which justifies the restriction to finite-state specifications and implementations.

Throughout this paper we work with Karp reductions, many-one reductions computable by deterministic Turing machines in polynomial time. This choice is justified since we reduce problems that are EXPTIME-complete.

3 Current Bounds

In [12], the three decision problems CI, C, and TR were studied for mixed and modal specifications. The results of [12] are summarized in Table 1. Two reductions were given in [12] that we appeal to here:

- a reduction of CI for modal specifications to C for *mixed* specifications
- a reduction of C for mixed specifications to TR for *mixed* specifications.

EXPTIME-hardness of CI for modal specifications would thus render EXPTIME-completeness of the decision problems CI, C, and TR for mixed specifications. We turn to this EXPTIME-hardness proof in the next section.

4 EXPTIME-Completeness Results

Theorem 4. *Let $\{(M_l, s_l)\}_{l \in \{1 \ldots k\}}$ be a finite family of modal specifications over the same action alphabet Σ. Deciding whether there exists an implementation (I, i) such that $(M_l, s_l) \prec (I, i)$ for all $l = 1 \ldots k$ is EXPTIME-hard.*

Table 1. A summary given in [12] of the results provided in [12].

	Modal specifications	Mixed specifications
Common impl.	PSPACE-hard, EXPTIME	PSPACE-hard, EXPTIME
Consistency	trivial	PSPACE-hard, EXPTIME
Thorough ref.	PSPACE-hard, EXPTIME	PSPACE-hard, EXPTIME

We prove Theorem 4 by demonstrating a PTIME reduction from $\mathrm{ATM}_{\mathrm{LB}}$. Given an ATM T and an input word w of length n we synthesize a collection of (pointed) modal specifications $\mathcal{M}_w^T = \{M_i \mid 1 \leq i \leq n\} \cup \{M_{\mathrm{head}}, M_{\mathrm{ctrl}}, M_{\mathrm{exist}}\}$ whose sum of sizes is polynomial in n and in the size of T, such that T accepts w iff there exists an (pointed) implementation I refining all members of $\mathcal{M}_w^T$.

Specifications M_i, M_{head}, M_{ctrl}, and M_{exist} model tape cell i, the current head position, the finite control of T, and acceptance (respectively). Common implementations of these specifications model action synchronization to agree on what symbol is read from the tape, what is the head position, what is the symbol written to the tape, in what direction the head moves, and what are the transitions taken by the finite control, and whether a computation is accepting. The achieved effect is that a common refinement of these specifications corresponds to an accepting computation of T on input w. More precisely, any common implementations will correspond to different unfoldings of the structure of the finite control into a computation tree based on the content of the tape cells and the tape head position.

We now describe the specifications in $\mathcal{M}_w^T$ both formally and through our running example in Figure 1. All specifications in $\mathcal{M}_w^T$ have the same alphabet[6]

$$\Sigma = \{\pi, \exists\} \cup (\Gamma \times \{1..n\} \times \Gamma \times \{\mathsf{l}, \mathsf{r}\})$$

where $\exists$ and π are fresh symbols whose transitions encode logical constraints like disjunction and conjunction. All other actions are of the form (a_1, i, a_2, d) and denote that the machine's head is over the ith cell of the tape, which contains the a_1 symbol, and that it shall be moved one cell in the direction d after writing a_2 in the current cell. The alphabet for our running example is

$$\{\pi, \exists\} \cup (\{0,1\} \times \{1..n\} \times \{0,1\} \times \{\mathsf{l}, \mathsf{r}\})$$

Encoding Tape Cells. For each tape cell i, specification M_i represents the possible contents of cell i. It has $|\Gamma|$ states $\{p_{\langle i,a\rangle}\}_{a\in\Gamma}$ and initial state $p_{\langle i,w_i\rangle}$, representing the initial contents of the ith cell. There are no must-transitions:

$$R^{\Box} = \emptyset$$

[6] A stricter and more complex reduction to CI of modal specifications over a *binary* alphabet is possible by encoding actions in binary form.

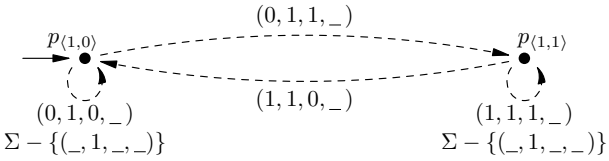


Fig. 3. Specification M_1 of the first tape cell in our running example, assuming $w_1 = 0$.

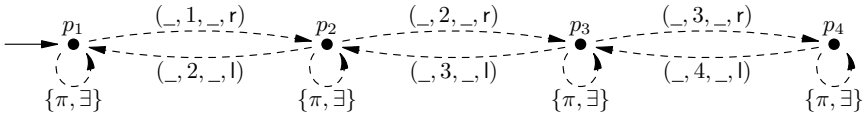


Fig. 4. Example of the head specification M_{head} assuming $|w| = 4$.

The may-transition relation connects any two states:

$$\text{for all symbols } a_1,\ a_2 \text{ in } \Gamma \text{ we have } (p_{\langle i,a_1\rangle}, (a_1, i, a_2, \_), p_{\langle i,a_2\rangle}) \in R^{\diamond}$$

Changes in cells other than i are also consistent with M_i:

$$\text{for all } a \in \Gamma \text{ if } i \neq j,\ 1 \leq j \leq n, \text{ then } (p_{\langle i,a\rangle}, (\_, j, \_, \_), p_{\langle i,a\rangle}) \in R^{\diamond}$$

Finally the π and $\exists$ actions may be used freely as they do not affect the contents of the cell:

$$(p_{\langle i,a\rangle}, \pi, p_{\langle i,a\rangle}) \in R^{\diamond} \text{ and } (p_{\langle i,a\rangle}, \exists, p_{\langle i,a\rangle}) \in R^{\diamond} \text{ for any } a \in \Gamma$$

There are no more may-transitions in M_i.

Figure 3 presents a specification M_1 for the leftmost cell of an ATM over a binary alphabet. In figures we visualize multiple transitions with the same source and target as single arrows labeled with sets of actions. Several labels placed by the same arrow denote a union of sets. Wildcards (the '_' symbol) are used to generate sets of actions that match the pattern in the usual sense.

Encoding The Head. Specification M_{head}, which tracks the current head position, has n states labeled p_1 to p_n — one for each possible position. Initially, the head occupies the leftmost cell, so p_1 is the initial state of M_{head}. There are no must-transitions:

$$R^{\Box} = \emptyset$$

May-transitions are consistent with any position changes based on the direction encoded in observed actions. More precisely,

$$\text{for every position } 1 \leq i < n \text{ we have } (p_i, (\_, i, \_, \mathsf{r}), p_{i+1}) \in R^{\diamond}$$

$$\text{for every } 1 < i \leq n \text{ we have } (p_i, (\_, i, \_, \mathsf{l}), p_{i-1}) \in R^{\diamond}$$

The π and $\exists$ transitions may again be taken freely, but in this case without moving the machine's head:

$$(p_i, \pi, p_i) \in R^{\diamond} \text{ and } (p_i, \exists, p_i) \in R^{\diamond} \text{ for each } 1 \leq i \leq n$$

There are no more may-transitions in M_{head}. Note that the head of T is only allowed to move between the first and nth cell in any computation. Figure 4 shows specification M_{head} for our running example.

Encoding The Finite Control. Specifications M_{ctrl} and M_{exist} model the finite control of the ATM T. Specification M_{exist} is indepenendent of the ATM T. It is defined in Figure 5. It ensures that a π-transition is taken after every $\exists$-transition. Specification M_{ctrl} mimics the finite control of T almost directly. Each control state $q_s \in Q$ is identified with a state in M_{ctrl} of the same name. Additional internal states of M_{ctrl} encode existential and universal branching:

for each q_s a state $q_{s\exists}$ with two $\exists$-transitions $(q_s, \exists, q_{s\exists}) \in R^{\diamond} \cap R^{\Box}$ is added

Dependent on $\mathsf{mode}(q_s)$, additional states and transitions are created:

- If $\mathsf{mode}(q_s) = \mathsf{Exst}$: for each $1 \leq i \leq n$, $a_{\text{old}} \in \Gamma$, and for each transition $(q_t, a_{\text{new}}, d) \in \delta(q_s, a_{\text{old}})$ add a may π-transition from $q_{s\exists}$ to a new intermediate state uniquely named $\langle q_s a_{\text{old}} i a_{\text{new}} d q_t \rangle$, and add a must-transition labeled $(a_{\text{old}}, i, a_{\text{new}}, d)$ from that intermediate state to q_t. Formally:

$$(q_{s\exists}, \pi, \langle q_s a_{\text{old}} i a_{\text{new}} d q_t \rangle) \in R^{\diamond}$$
$$(\langle q_s a_{\text{old}} i a_{\text{new}} d q_t \rangle, (a_{\text{old}}, i, a_{\text{new}}, d), q_t) \in R^{\diamond} \cap R^{\Box}$$

  Figure 6 shows this encoding for the state e of our running example.
- If $\mathsf{mode}(q_s) = \mathsf{Univ}$: for each $1 \leq i \leq n$, $a_{\text{old}} \in \Gamma$, and for each transition $(q_t, a_{\text{new}}, d) \in \delta(q_s, a_{\text{old}})$ add a may π-transition from $q_{s\exists}$ to an intermediate state named $\langle q_s a_{\text{old}} i \rangle$, and add a must-transition labeled $(a_{\text{old}}, i, a_{\text{new}}, d)$ from the intermediate state $\langle q_s a_{\text{old}} i \rangle$ to q_t. Formally:

$$(q_{s\exists}, \pi, \langle q_s a_{\text{old}} i \rangle) \in R^{\diamond} \quad , \qquad (\langle q_s a_{\text{old}} i \rangle, (a_{\text{old}}, i, a_{\text{new}}, d), q_t) \in R^{\diamond} \cap R^{\Box}$$

The initial state of M_{ctrl} is its state named q_0, where q_0 is the initial state of T. Figure 7 demonstrates the encoding of the state u_1 of the ATM in Figure 1. The complete specification M_{ctrl} for our running example is shown in Figure 8.

Notice how the two specifications M_{ctrl} and M_{exist} cooperate to enforce the nature of alternation. For example, for an existential state, M_{ctrl} forces every implementation to have an $\exists$-transition, which may be followed by a π-transition. Simultaneously M_{exist} allows an $\exists$-transition but requires a π-transition. Effectively at least one of the π branches from M_{ctrl} must be implemented (which is an encoding of a disjunction).

The complete family of specifications $\mathcal{M}_w^T$ contains all the specifications described above:

$$\mathcal{M}_w^T = \{M_i \mid 1 \leq i \leq n\} \cup \{M_{\text{head}}, M_{\text{ctrl}}, M_{\text{exist}}\}$$

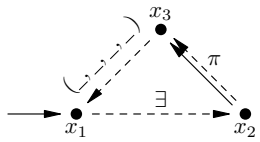


Fig. 5. Specification M_{exist} enforces a π-transition after each $\exists$-transition.

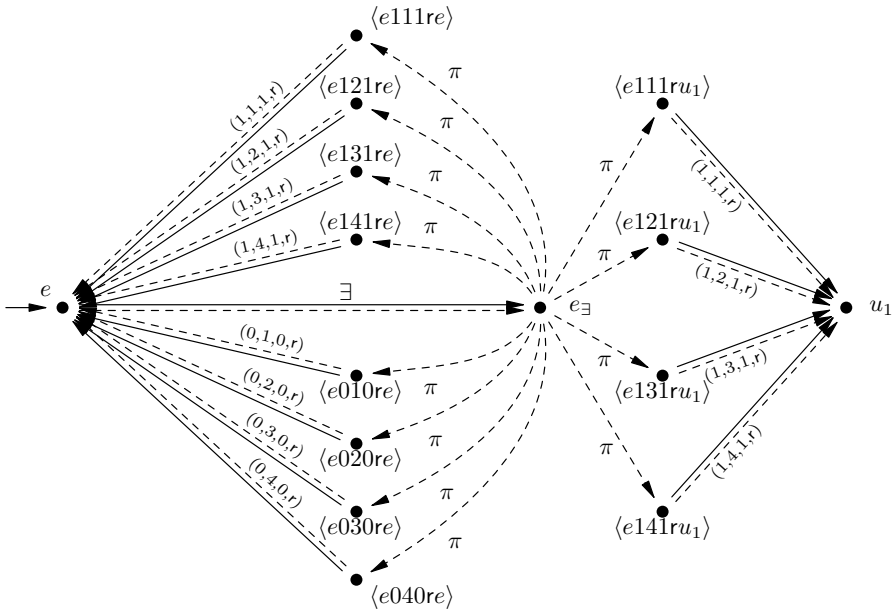


Fig. 6. Encoding for the existential state of the running example, assuming $|w| = 4$.

All these specifications are modal by construction. Since the sum of their sizes is bounded by a polynomial in n and in the size of T, it remains to prove the following lemma:

Lemma 5. *For each linearly bounded ATM T and an input w, T accepts w iff the set of modal specifications $\mathcal{M}_w^T$ has a common implementation.*

The proof of Lemma 5 will appear in the final version of the paper. We mention here some points of interest. From an accepting computation tree $\mathsf{T}_{\langle T,w\rangle}$ one can construct a specification N by structural induction on $\mathsf{T}_{\langle T,w\rangle}$. This N effectively adds to $\mathsf{T}_{\langle T,w\rangle}$ some new states and labeled transitions so that the computation encoded in $\mathsf{T}_{\langle T,w\rangle}$ then interlocks with the action synchronization of specifications in $\mathcal{M}_w^T$. Since N is of the form (S, R, R) it suffices to show that N is a common refinement of all members in $\mathcal{M}_w^T$. This is a lengthy but routine argument.

For the converse, a common implementation of $\mathcal{M}_w^T$ is cycle-free by our assumption that T never repeats a configuration. So that pointed common implementation is a DAG and we use structural induction on that DAG to synthesize an accepting computation tree of T for input w. This makes use of the fact that the head of T never reaches a cell that was not initialized by input w.

Further results. Theorem 4 states EXPTIME-hardness of CI for *modal* specifications. Together with the upperbound given in [12] we conclude that this bound is tight: CI is EXPTIME-complete. Moreover, by applying the reduction of CI for modal specifications to C for mixed specifications [12] we conclude that C for mixed specifications is EXPTIME-complete. Furthermore by appealing to the reduction of C for mixed specifications to TR for mixed specifications [12], we obtain that TR for mixed specifications is EXPTIME-complete as well.

Corollary 6. *The complexities shown in Table 2 are correct.*

5 Discussion

One complexity gap remains in Table 2, that for thorough refinement of *modal* specifications. Despite having made an extensive effort we can presently show neither
EXPTIME-hardness nor membership in PSPACE for this problem.

Table 2. Tabular summary of the results provided in this paper (in bold).

	Modal specifications	Mixed specifications
Common impl.	**EXPTIME-complete**	**EXPTIME-complete**
Consistency	trivial [1]	**EXPTIME-complete**
Thorough ref.	PSPACE-hard, EXPTIME [12]	**EXPTIME-complete**

In this context, it is useful to state that thorough refinement can be reduced to certain validity checks. First, as observed in [12], mixed and modal specifications (M, s) have characteristic formulæ $\Psi_{(M,s)}$ [25] in the modal μ-calculus such that pointed labeled transition systems (L, l) are implementations of (M, s) iff (L, l) satisfies $\Psi_{(M,s)}$. This was already observed in [2] for such formulæ written in vectorized form. So the thorough refinement problem of whether $(M, s) \prec_{th} (N, t)$ reduces to a validity check of $\neg\Psi_{(N,t)} \vee \Psi_{(M,s)}$. This raises the question of whether the validity problem for formulae given in the vectorized form of [2] is in PSPACE or whether it is EXPTIME-hard; that problem is known to be in EXPTIME (see for example [12]).

Second, we can reduce thorough refinement to a universal version of generalized model checking [13]. In loc. cit. Bruns and Godefroid consider judgments $\mathsf{GMC}(M, s, \varphi)$ which are true iff there exists an implementation of (M, s) satisfying φ. They remark that this generalizes both model checking (when (M, s) is an implementation) and satisfiability checking (when (M, s) is such that all labeled transition systems refine it). This existential judgment has a universal dual (see e.g. [26]), $\mathsf{VAL}(M, s, \varphi)$ which is true iff all implementations of (M, s) satisfy φ, thus generalizing model checking and validity checking. The former judgment is useful for finding counter-examples, the latter one for verification; e.g. both uses can be seen in the CEGAR technique for program verification of [27]. Since $(M, s) \prec_{th} (N, t)$ directly reduces to $\mathsf{VAL}(N, t, \Psi_{(M,s)})$, it would be of interest to understand the exact complexity of $\mathsf{VAL}(N, t, \varphi)$ for modal specifications (N, t) when φ ranges over characteristic formulæ $\Psi_{(M,s)}$ in vectorized form.

We remark that by translations and completeness results presented in [28] it follows that all complexity bounds presented here carry over to partial Kripke structures and Kripke modal transition systems.

6 Conclusion

We have discussed three fundamental decision problems for modal and mixed specifications: common implementation, consistency, and thorough refinement. For *modal* specifications, consistency is trivially true, while thorough refinement was previously shown to be PSPACE-hard and in EXPTIME [12]. For the remaining decision problems we have shown here that they are all EXPTIME-complete in the sum of the sizes of mixed or modal specifications.

We have appealed to known reductions between some of these problems [12] and, crucially, to a new reduction of input acceptance for linearly bounded alternating Turing machines to the existence of a common implementation for modal specifications – sketched in this extended abstract. The exact complexity of thorough refinement for modal specifications is subject to further investigation.

References

1. Larsen, K.G., Thomsen, B.: A modal process logic. In: Third Annual IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society (1988)

2. Larsen, K.G.: Modal specifications. In Sifakis, J., ed.: Automatic Verification Methods for Finite State Systems. Volume 407 of Lecture Notes in Computer Science., Springer (1989) 232–246
3. Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: Fifth Annual IEEE Symposium on Logics in Computer Science (LICS), 4–7 June 1990, Philadelphia, PA, USA. (1990) 108–117
4. Cerans, K., Godskesen, J.C., Larsen, K.G.: Timed modal specification - theory and tools. In: CAV '93: Proceedings of the 5th International Conference on Computer Aided Verification, London, UK, Springer-Verlag (1993) 253–267
5. Larsen, K.G., Steffen, B., Weise, C.: A constraint oriented proof methodology based on modal transition systems. In: Tools and Algorithms for Construction and Analysis of Systems. (1995) 17–40
6. Larsen, K.G., Steffen, B., Weise, C.: Fischer's protocol revisited: a simple proof using modal constraints. Lecture Notes in Computer Science **1066** (1996) 604–615
7. Dams, D.: Abstract Interpretation and Partition Refinement for Model Checking. PhD thesis, Eindhoven University of Technology (July 1996)
8. Dams, D., Gerth, R., Grumberg, O.: Abstract interpretation of reactive systems. ACM Trans. Program. Lang. Syst. **19**(2) (1997) 253–291
9. Larsen, K.G., Nyman, U., Wąsowski, A.: On modal refinement and consistency. In Caires, L., Vasconcelos, V.T., eds.: CONCUR. Volume 4703 of Lecture Notes in Computer Science., Springer (2007) 105–119
10. Larsen, K.G., Nyman, U., Wąsowski, A.: Modal I/O automata for interface and product line theories. In Nicola, R.D., ed.: ESOP. Volume 4421 of Lecture Notes in Computer Science., Springer (2007) 64–79
11. Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wąsowski, A.: 20 years of modal and mixed specifications. Bulletin of EATCS (94) (June 2008) Available at `http://processalgebra.blogspot.com/2008/05/concurrency-column-for-beatcs-june-2008.html`.
12. Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wąsowski, A.: Complexity of decision problems for mixed and modal specifications. In: FoSSaCS'08. Volume 4962 of Lecture Notes in Computer Science., Springer (2008)
13. Bruns, G., Godefroid, P.: Generalized model checking: Reasoning about partial state spaces. In Palamidessi, C., ed.: CONCUR. Volume 1877 of Lecture Notes in Computer Science., Springer (2000) 168–182
14. Hussain, A., Huth, M.: On model checking multiple hybrid views. Technical report, Department of Computer Science, University of Cyprus (2004) TR-2004-6.
15. Hussain, A., Huth, M.: Automata games for multiple-model checking. Electr. Notes Theor. Comput. Sci. **155** (2006) 401–421
16. Fischbein, D., Uchitel, S., Braberman, V.: A foundation for behavioural conformance in software product line architectures. In: ROSATEA '06 Proceedings, New York, NY, USA, ACM Press (2006) 39–48
17. Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. J. ACM **28**(1) (1981) 114–133
18. Laroussinie, F., Sproston, J.: State explosion in almost-sure probabilistic reachability. Inf. Process. Lett. **102**(6) (2007) 236–241
19. Sipser, M.: Introduction to the Theory of Computation. International Thomson Publishing (1996)
20. Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM Trans. Program. Lang. Syst. **16**(5) (1994) 1512–1542

21. Park, D.: Concurrency and automata on infinite sequences. In: Proceedings of the 5th GI-Conference on Theoretical Computer Science, London, UK, Springer-Verlag (1981) 167–183
22. Hüttel, H.: Operational and denotational properties of modal process logic. Master's thesis, Computer Science Department. Aalborg University (1988)
23. Xinxin, L.: Specification and Decomposition in Concurrency. PhD thesis, Department of Mathematics and Comnputer Science, Aalborg University (April 1992)
24. Schmidt, H., Fecher, H.: Comparing disjunctive modal transition systems with a one-selecting variant. Submitted for publication (2007)
25. Huth, M.: Labelled transition systems as a Stone space. Logical Methods in Computer Science **1**(1) (January 2005) 1–28
26. Antonik, A., Huth, M.: On the complexity of semantic self-minimization. In: Proc. AVOCS 2007. To appear in ENTCS.
27. Godefroid, P., Huth, M.: Model checking vs. generalized model checking: Semantic minimizations for temporal logics. In: LICS, IEEE Computer (2005) 158–167
28. Godefroid, P., Jagadeesan, R.: On the expressiveness of 3-valued models. In Zuck, L.D., Attie, P.C., Cortesi, A., Mukhopadhyay, S., eds.: VMCAI. Volume 2575 of Lecture Notes in Computer Science., Springer (2003) 206–222

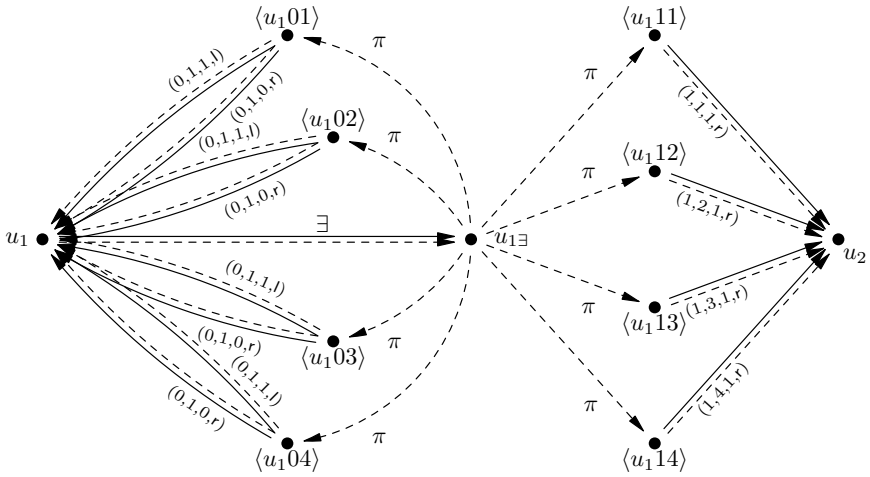


Fig. 7. Encoding for the universal state u_1 of the running example, assuming $|w| = 4$.

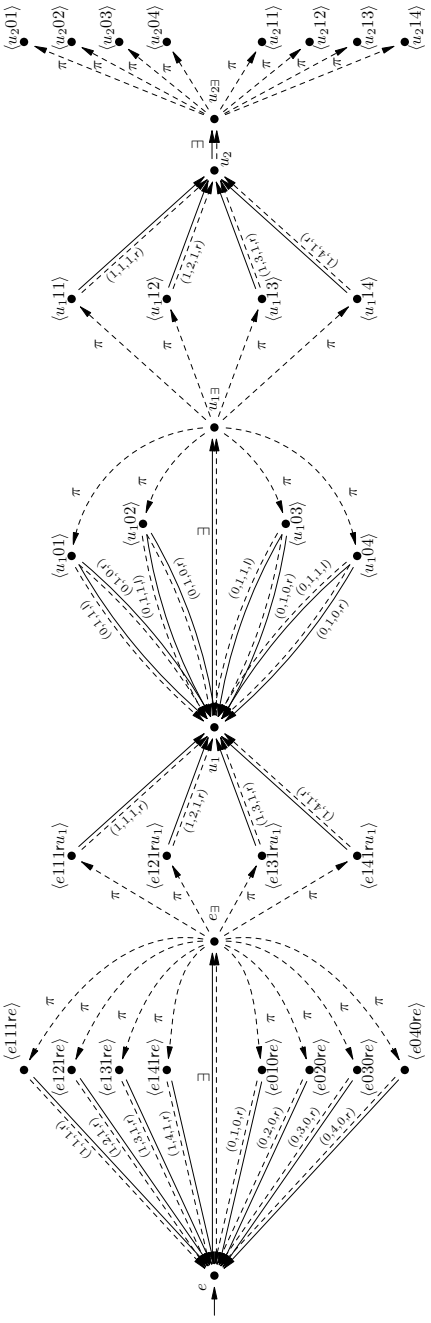


Fig. 8. The entire specification M_{ctrl} for the example of Figure 1 assuming $|w| = 4$.

A Proof of Lemma 5

We need to argue that if the linearly bounded ATM T has an accepting computation on input w, then the set $\mathcal{M}_w^T$ of constructed modal specifications will have a common implementation; and, conversely, that if this set $\mathcal{M}_w^T$ of modal specifications has a common implementation, then this common implementation witnesses an accepting computation for the linearly bounded ATM T on input w. We will prove each of these directions separately.

A.1 Acceptance implies existence of common implementation

Let the ATM T accept input w. We mean to show that $\mathcal{M}_w^T$ has a common implementation. Since we have assumed that T does not repeat configurations on any computation path, we know that there exists a computation tree $\mathsf{T}_{\langle T,w\rangle}$ demonstrating that T accepts w in an exponentially bounded number of steps.

Construction of modal specification N. From $\mathsf{T}_{\langle T,w\rangle}$ we can construct a modal specification

$$N = (N_{\text{states}}, R_N, R_N)$$

over Σ, where N_{states} is a set of states, R_N is a transition relation and Σ is the alphabet of specifications in $\mathcal{M}_w^T$. The argument that N refines all specifications of $\mathcal{M}_w^T$ will follow shortly after the construction.

Since N has identical must- and may transition relation we shall just refer to transitions for N without mentioning their type. N has three kinds of states:

- type 1 states *not* subscripted with special actions, for example $n_{\langle q_0,1,w\rangle}$
- type 2 states with an extra subscript $\exists$, as in state $n_{\langle q,i,\tau\rangle\exists}$
- type 3 states with an extra subscript π, as in state $n_{\langle q,i,\tau\rangle\pi}$

We construct N in a recursive manner starting from the root of the accepting computation tree. We start by creating the initial state of N labelled $n_{\langle q_0,1,w\rangle}$, where $\langle q_0, 1, w\rangle$ is the configuration of the root node in $\mathsf{T}_{\langle T,w\rangle}$. We shall be adding new successor states and transitions in a top-down fashion while progressing. Our recursive procedure accepts two parameters $(\langle q, i, \tau\rangle, n_{\langle q,i,\tau\rangle})$: a node from $\mathsf{T}_{\langle T,w\rangle}$ and a state from N_{states}. For any pair of parameters $(\langle q, i, \tau\rangle, n_{\langle q,i,\tau\rangle})$ proceed as follows:

- If $\mathsf{mode}(q) = \mathsf{Univ}$ create two new states $n_{\langle q,i,\tau\rangle\exists}$ and $n_{\langle q,i,\tau\rangle\pi}$ and a $\exists$-transition from $n_{\langle q,i,\tau\rangle}$ to $n_{\langle q,i,\tau\rangle\exists}$, and a π-transition from $n_{\langle q,i,\tau\rangle\exists}$ to $n_{\langle q,i,\tau\rangle\pi}$. Second, for each of the successors $\langle q', i', \tau'\rangle$ of $\langle q, i, \tau\rangle$ create a new state $n_{\langle q',i',\tau'\rangle}$ and a transition from $n_{\langle q,i,\tau\rangle\pi}$ to $n_{\langle q',i',\tau'\rangle}$ labelled by (τ_i, i, τ_i', d) where $d = \mathsf{r}$ if $i' = i + 1$ and $d = \mathsf{l}$ otherwise. Then continue recursively for every successor $\langle q', i', \tau'\rangle$ of $\langle q, i, \tau\rangle$, and its corresponding state $n_{\langle q',i',\tau'\rangle}$.

– If $\mathsf{mode}(q) = \mathsf{Exst}$ create two new states $n_{\langle q,i,\tau\rangle\exists}$ and $n_{\langle q,i,\tau\rangle\pi}$ and an $\exists$-transition from $n_{\langle q,i,\tau\rangle}$ to $n_{\langle q,i,\tau\rangle\exists}$ and a π-transition from $n_{\langle q,i,\tau\rangle\exists}$ to $n_{\langle q,i,\tau\rangle\pi}$. Second, because $\mathsf{T}_{\langle T,w\rangle}$ is accepting, we know that there exists at least one successor configuration $\langle q', i', \tau'\rangle$ for which the subtree with this configuration as root accepts. Select this one configuration and create a new state $n_{\langle q',i',\tau'\rangle}$ and a transition from $n_{\langle q,i,\tau\rangle\pi}$ to $n_{\langle q',i',\tau'\rangle}$ labelled by (τ_i, i, τ'_i, d) where $d = \mathsf{r}$ if $i' = i+1$ and $d = \mathsf{l}$ otherwise. Then continue recursively with $\langle q', i', \tau'\rangle$ and $n_{\langle q',i',\tau'\rangle}$.

Observe that the above recursive computation terminates in universal states with no successors, due to an iteration over an empty set. This is so since $\mathsf{T}_{\langle T,w\rangle}$ is an accepting computation tree and so we are guaranteed that the existential branch can always continue, and, because T only allows execution of a bounded number of steps, every branch of the above recursive procedure will eventually terminate.

Proof that N refines all specifications in $\mathcal{M}_w^T$. We shall now prove that specification $(N, n_{\langle q_0,1,w\rangle})$ refines each of the modal specifications in $\mathcal{M}_w^T$. In the following we write τ_i, meaning the ith symbol of the tape state τ.

1. $(M_{\text{exist}}, x_1) \prec (N, n_{\langle q_0,1,w\rangle})$: Recall that the specification M_{exist} has exactly three states named x_1, x_2 and x_3 (see Figure 5). Consider the following binary relation on states of M_{exist} and states of N:

$$\begin{aligned} Q_1 = \{(x_1, n_{\langle q_s,i,\tau\rangle}) \mid n_{\langle q_s,i,\tau\rangle} \in N_{\text{states}}\} \;\cup \\ \{(x_2, n_{\langle q_s,i,\tau\rangle\exists}) \mid n_{\langle q_s,i,\tau\rangle\exists} \in N_{\text{states}}\} \;\cup \\ \{(x_3, n_{\langle q_s,i,\tau\rangle\pi})) \mid n_{\langle q_s,i,\tau\rangle\pi} \in N_{\text{states}}\} \;. \end{aligned}$$

   We shall argue that Q_1 witnesses a refinement of (M_{exist}, x_1) by $(N, n_{\langle q_0,1,w\rangle})$. First, observe that the pair of initial states $(x_1, n_{\langle q_0,1,w\rangle})$ of M_{exist} and N are related in Q_1. Second, check that Q_1 fulfils the conditions of Definition 2:

Def. 2(1) We want to show for all pairs $(x, n) \in Q_1$ that for all states x' of M_{exist} if $(x, a, x') \in R^{\Box}_{M_{\text{exist}}}$ then there exists a state $n' \in N_{\text{states}}$ with $(n, a, n') \in R^{\Box}_N$ and $(x', n') \in Q_1$. A must-transition occurs in $R^{\Box}_{M_{\text{exist}}}$ only if $x = x_2$. In this case there is exactly one must π-transition going to x_3. We see from Q_1 that x_2 is paired only with states of form $n = n_{\langle q_s,i,\tau\rangle\exists}$. By construction of N, the latter state always has a must π-transition to some state $n' = n_{\langle q_s,i,\tau\rangle\pi}$ which gives us that $(x', n') \in Q_1$ by the construction of Q_1.

Def. 2(2) We want to show for all pairs $(x, n) \in Q_1$ that for all states n' of N_{states} if $(n, a, n') \in R^{\Diamond}_N$ then there exists a state x' of M_{exist} such that $(x, a, x') \in R^{\Diamond}_{M_{\text{exist}}}$ with $(x', n') \in Q_1$. These argument is split into three sub-cases.

- If n is of type 1, $n = n_{\langle q_s,i,\tau\rangle}$, then by Q_1's construction $x = x_1$. By construction of N any may-transition leaving n will be labelled by $\exists$ and target a type 2 state $n' = n_{\langle q_s,i,\tau\rangle\exists}$. This can be matched by $(x_1, \exists, x_2) \in R^{\Diamond}_{M_{\text{exist}}}$ and again gives us $(x', n') \in Q_1$ by construction of Q_1.
- If n is of type 2, $n = n_{\langle q_s,i,\tau\rangle\exists}$, then by Q_1's construction $x = x_2$. By construction of N there is exactly one may π-transition leaving n. It targets a state n' of type 3, so $n' = n_{\langle q_s,i,\tau\rangle\pi}$. This can be matched by $(x_2, \pi, x_3) \in R^{\Diamond}_{M_{\text{exist}}}$ and gives us $(x', n') \in Q_1$ by construction of Q_1.
- If n is of type 3, $n = n_{\langle q_s,i,\tau\rangle\pi}$, then by Q_1's construction $x = x_3$. By construction of N all possible may-transitions leaving n target type 1 states of the form $n' = n_{\langle q_s,i,\tau\rangle}$. All these transitions have labels in $(\_,\_,\_,\_)$. These can all be matched by (M_{exist}, x_3), as that specification contains all transitions of type $(\_,\_,\_,\_)$ going from x_3 to x_1. Since x_1 is paired with all states of type 1 in Q_1 this again gives us that $(x', n') \in Q_1$.

2. For each cell $1 \leq i \leq n$ show that $(M_i, p_{\langle i,w_i\rangle}) \prec (N, n_{\langle q_0,1,w\rangle})$. For any selection of i above consider the following relation Q_2^i over the states of M_i and the states of N.

$$Q_2^i = \{(p_{\langle i,\tau_i\rangle}, n) \mid n = n_{\langle q_s,j,\tau\rangle} \text{ or } n = n_{\langle q_s,j,\tau\rangle\pi} \text{ or } n = n_{\langle q_s,j,\tau\rangle\exists}\} \ .$$

   The first step is to see that the initial states of the two specifications are related in Q_2^i. This is clearly the case since the initial state of each M_i is set to be the state $p_{\langle i,w_i\rangle}$ that matches the content of the corresponding cell of the input tape. After this we need to show that given $(p, n) \in Q_2^i$ the refinement condition is preserved.

Def. 2(1) This condition is vacuously true since each M_i has no must transitions.

Def. 2(2) We want to show for all pairs $(p, n) \in Q_2^i$ that for all states $n' \in N_{\text{states}}$ if $(n, a, n') \in R^{\Diamond}_N$ then there exists a state p' of M_i such that $(p, a, p') \in R^{\Diamond}_{M_i}$ with $(p', n') \in Q_2^i$. With only one exception, whenever N takes a may-transition M_i will be able to match it. The exception is if the label contains as its old tape symbol, a symbol different from the one that M_i has in its current state and where i is the current position of the head in n, so $i = j$. Since the transitions of N are created from a legal computation tree for the ATM T we can conclude that N will never change the content of the tape without writing to it and N will thus never try to read something from a tape cell that is not in that given tape cell. It will also always update the new content of the tape cell correctly and we are thus assured that $(p', n') \in Q_2^i$.

3. Show that $(M_{\text{head}}, p_1) \prec (N, n_{\langle q_0,1,w\rangle})$: The relation Q_3 witnessing this refinement is defined as follows:

$$Q_3 = \{\, (p_i, n) \mid n = n_{\langle q_s,i,\tau\rangle} \text{ or } n = n_{\langle q_s,i,\tau\rangle\pi} \text{ or } n = n_{\langle q_s,i,\tau\rangle\exists} \,\} \ .$$

We first have to ensure that the initial states of the two specifications are in Q_3. This is clearly the case since the initial state of N will have $i = 1$ and this will be related to p_1 which is the initial state of M_{head}. Second, we need to show, that for any given $(p, n) \in Q_3$ the two refinement conditions of Definition 2 are preserved.

Def. 2(1) This condition is vacuously satisfied since M_{head} has no must transitions.

Def. 2(2) We need to show that whenever $(n, a, n') \in R_N^{\diamond}$ then there exists p', a state of M_{head}, such that $(p, a, p') \in R_{M_{\text{head}}}^{\diamond}$ with $(p', n') \in Q_3$. We just discuss the case when n is of type 3 here, so $n = n_{\langle q_s,i,\tau\rangle\pi}$. For the remaining two types the transitions leaving n do not move the head and the preservation of refinement can be concluded directly.
By construction of N whenever $n_{\langle q_s,i,\tau\rangle\pi}$ takes a may-transition then this transition is labeled $(\_, i, \_, d)$ targeting a type 1 state $n_{\langle q',i,'\rangle\tau'}$, where $i' = i+1$ if $d = \mathsf{r}$ and $i' = i-1$ otherwise. Now by construction of M_{head} the state p_i can match such a transition moving to $p_{i'}$ accordingly. The only case where M_{head} would not be able to match is when N would try to move the head off either end of the tape, but this will never happen since N is constructed from a legal accepting computation tree. Thus we conclude that the refinement condition is preserved.

4. $(M_{\text{ctrl}}, q_0) \prec (N, n_{\langle q_0,1,w\rangle})$: Consider the following binary relation Q_4 on states of M_{ctrl} and N:

$$\begin{aligned} Q_4 = \{&(q_s, n) \mid n = n_{\langle q_s,i,\tau\rangle}\} \cup \\ &\{(q_{s\exists}, n) \mid n = n_{\langle q_s,i,\tau\rangle\exists}\} \cup \\ &\{(\langle q_s \tau_i i\rangle, n_{\langle q_s,i,\tau\rangle\pi}) \mid \mathsf{mode}(q_s) = \mathsf{Univ}\} \ \cup \\ &\{(\langle q_s \tau_i i a_2 d q_t\rangle, n_{\langle q_s,i,\tau\rangle\pi}) \mid \mathsf{mode}(q_s) = \mathsf{Exst} \text{ and} \\ &\qquad (n_{\langle q_s,i,\tau\rangle\pi}, (\tau_i, i, a_2, d), n_{\langle q_t,i',\tau'\rangle}) \in R_N^{\diamond}\} \ . \end{aligned}$$

First, observe that the initial states of the two specifications are in Q_4, as q_0 is the initial state of M_{ctrl} and $n_{\langle q_0,1,w\rangle}$ is the initial state of N (see the first summand in the definition of Q_4). Secondly, we need to show that, given a pair $(q, n) \in Q_4$, the two refinement conditions of Definition 2 are preserved.

Def. 2(1) We need to show that whenever $(q, a, q') \in R_{M_{\text{ctrl}}}^{\Box}$ then there exists a state $n' \in N_{\text{states}}$ such that $(n, a, n') \in R_N^{\Box}$ with $(q', n') \in Q_4$. The argument is split in four cases.

- If $q = q_s$ for some $q_s \in Q$ (a state of the ATM T) then there is exactly one must $\exists$-transition leaving it, which targets $q_{s\exists}$. This transition can be matched by an $\exists$-transition leaving $n_{\langle q_s,i,\tau\rangle}$ and targeting $n_{\langle q_s,i,\tau\rangle\exists}$. These new target states remain in relation Q_4, as per the above definition.

- If $q = q_{s\exists}$ for some $q_s \in Q$ (a state of the ATM T) then the condition is satisfied vacuously. There is simply no must-transition leaving q.
- If q has the form $\langle q_s \tau_i i \rangle$, where q_s is a universal state of the ATM T, then n has the form $n_{\langle q_s,i,\tau\rangle\pi}$. But since $n_{\langle q_s,i,\tau\rangle\pi}$ was constructed by our recursive procedure from a universal configuration of an accepting computation tree we know that, for all must-transitions leaving $\langle q_s \tau_i i \rangle$ to some state q_t, there will be a matching must-transition in N leaving $n_{\langle q_s,i,\tau\rangle\pi}$ and targeting $n_{\langle q_t,i',\tau'\rangle}$, which is in relation with q_t as per the first summand in the definition of Q_4.
- If q has the form $\langle q_s \tau_i i a_2 d q_t \rangle$, where q_s is an existential state of the ATM T, then n has the form $n_{\langle q_s,i,\tau\rangle\pi}$. The state $\langle q_s \tau_i i a_2 d q_t \rangle$ has exactly one must-transition labeled (τ_i, i, a_2, d) and targeting state q_t. Since q_s is an existential state, we know that $n_{\langle q_s,i,\tau\rangle\pi}$ was constructed from an existential configuration and consequently there is a single must-transition leaving it. This transition is labeled (τ_i, i, a_2, d) as per construction of the Q_4 relation (see the last summand). Finally this transition targets $n' = n_{\langle q_t,i',\tau'\rangle}$. And thus we again have that $(q', n') \in Q_4$.

Def. 2(2) We want to show that if $(n, a, n') \in R_N^\diamond$ then there exists a state q' of M_{ctrl} such that $(q, a, q') \in R_{M_{\text{ctrl}}}^\diamond$ with $(q', n') \in Q_4$. We split the argument into three cases based on the type of state n.

- If n is of type 1, so $n = n_{\langle q_s,i,\tau\rangle}$ then by construction of N there is a may $\exists$-transition leaving n targeting $n_{\langle q_s,i,\tau\rangle\exists}$. This is followed by $(q_s, \exists, q_{s\exists}) \in R_{M_{\text{ctrl}}}^\diamond$ and again gives us that $(q', n') \in Q_4$.
- If n is of type 2, $n = n_{\langle q_s,i,\tau\rangle\exists}$ then by the construction of Q_4 (see the second summand) q is of the form $q_{s\exists}$. By the construction procedure of N there is a single may π-transition leaving $n_{\langle q_s,i,\tau\rangle\exists}$ and targeting $n' = n_{\langle q_s,i,\tau\rangle\pi}$.
  - If $\mathsf{mode}(q_s) = \mathsf{Univ}$, then there is exactly one transition $(q_{s\exists}, \pi, \langle q_s \tau_i i \rangle) \in R_{M_{\text{ctrl}}}^\diamond$; its target state is related to $n_{\langle q_s,i,\tau\rangle\pi}$ in Q_4.
  - If $\mathsf{mode}(q_s) = \mathsf{Exst}$ then there can be many may π-transitions leaving $q_{s\exists}$. We will choose which one to match with, based on the label of the single transition leaving $n_{\langle q_s,i,\tau\rangle\pi}$. We are, so to speak, looking one step ahead. Since $n_{\langle q_s,i,\tau\rangle\pi}$ says that the head is in position i over a tape containing τ we choose to match our transition with the transition of M_{ctrl} targeting the state whose name matches the prefix "$\langle q_s \tau_i i$". Such a state always exists by construction of M_{ctrl} and it is exactly the state which is related to $n_{\langle q_s,i,\tau\rangle\pi}$ in Q_4 (see the last summand).
- For n of type 3, so $n = n_{\langle q_s,i,\tau\rangle\pi}$, we split the argument into two cases based on the mode of q_s in the ATM T.
  - First, if $\mathsf{mode}(q_s) = \mathsf{Univ}$ then there are possibly several may-transitions leaving $n_{\langle q_s,i,\tau\rangle\pi}$. Since N has been created from a legal computation tree, we know that any may transition leaving

$n_{\langle q_s,i,\tau\rangle\pi}$ and targeting $n' = n_{\langle q_t,i',\tau'\rangle}$ follows the transition relation δ of T. Moreover, by construction of M_{ctrl}, its state $\langle q_s\tau_i i\rangle$ will consequently be able to match this transition arriving in the state q_t related to n' in Q_4.

- Second, if $\mathsf{mode}(q_s) = \mathsf{Exst}$ then there is exactly one may transition leaving $n_{\langle q_s,i,\tau\rangle\pi}$ and exactly one may-transition leaving $s = \langle q_s\tau_i i a_2 d q_t\rangle$. These transitions have the same label and have respective target states $n_{\langle q_t,i',\tau'\rangle}$ and q_t, which are related in Q_4.

This concludes the argument that each specification in $\mathcal{M}_w^T$ is refined by N.

A.2 Existence of common implementation implies acceptance

Let $\mathcal{M}_w^T$ have a common implementation. We need to show that the ATM T accepts input w. Given a modal specification

$$U_{\text{new}} = (U_{\text{states}}, R_U, R_U)$$

that is a common implementation of $\mathcal{M}_w^T$ we will construct a computation tree $\mathsf{T}_{\langle M,w\rangle}$ demonstrating that T accepts w.

Since U_{new} is a common implementation of $\mathcal{M}_w^T$ we have $3 + n$ refinement relations:

$$Q_{\text{ctrl}},\ Q_{\text{tape}},\ Q_{\text{exist}},\ Q_1,\ \ldots,\ Q_n$$

— each demonstrating for one of the corresponding specifications $S \in \mathcal{M}_w^T$ that $S \prec U_{\text{new}}$. We construct a computation tree $\mathsf{T}_{\langle M,w\rangle}$, which witnesses that T accepts w, by structural induction, presenting the construction itself and a proof of its correctness simultaneously. The induction hypothesis is:

> **IH:** For every configuration $\langle q, m, \tau\rangle$ of $\mathsf{T}_{\langle M,w\rangle}$ the following conditions (1) and (2) hold:
>
> $$\begin{aligned}&\exists u_x \in U_{\text{states}}\colon\\ &(u_x, x_1) \in Q_{\text{exist}} \ \text{ and}\\ &(u_x, q) \in Q_{\text{ctrl}} \ \text{ and}\\ &(u_x, p_m) \in Q_{\text{tape}} \ \text{ and}\\ &\forall k \in \{1,\ldots,n\}\colon (u_x, p_{m,\tau_k}) \in Q_k \end{aligned} \tag{1}$$
>
> (2) Moreover if a configuration $\langle q', m', \tau'\rangle$ is a successor of $\langle q, m, \tau\rangle$ in $\mathsf{T}_{\langle M,w\rangle}$ then it also is a successor of $\langle q, m, \tau\rangle$ in the ATM T, and conversely $\mathsf{T}_{\langle M,w\rangle}$ has all the successors of $\langle q, m, \tau\rangle$ that T has for universal states and at least one of them for all existential states.

Notice that the name q above is used in two meanings: as a control state in ATM T (as in $\langle q, m, \tau\rangle$) and as a (corresponding) state in specification M_{ctrl} in the middle line of (1).

Base case: The base case consists of showing that the initial state $\langle q_0, 1, w\rangle$ of the computation tree $\mathsf{T}_{\langle M,w\rangle}$ fulfills the induction hypothesis.

U_{new} has a distinct initial state u_0 and we know that since $M_{\text{ctrl}} \prec U_{\text{new}}$ there is a pair $(u_0, q_0) \in Q_{\text{ctrl}}$ fulfilling line two of the induction hypothesis. Since $M_{\text{head}} \prec U_{\text{new}}$ we know that $(u_0, p_1) \in Q_{\text{tape}}$ fulfilling line four of the induction hypothesis. Since w is the initial content of the tape we know that, for each relation, there exists a pair (u_0, p_{1,w_k}) relating u_0 to the initial state of the relevant M_k. We also have that $(u_0, x_1) \in Q_{\text{exist}}$ finishing the base case.[7]

Inductive step: Assuming that the induction hypothesis holds for the current state $\langle q, m, \tau\rangle$, we now want to show that we can construct the next level of $\mathsf{T}_{\langle M,w\rangle}$ in such a way that the induction hypothesis holds for all its successors.

Before we split into two cases based on modes of states, we shall describe the part of the proof which these two have in common. The induction hypothesis allows us to assume existence of a specific state u_x of U_{states} and refinement relations showing that the induction hypothesis holds. Since u_x is related to a state without a π of $\exists$ subscript in M_{ctrl}, that u_x must implement a $\exists$ transition to a new state, let us call this state $u_{x\pi}$. Because $(u_x, x_1) \in Q_{\text{exist}}$ we know that $(u_{x\pi}, x_2) \in Q_{\text{exist}}$ and thus $u_{x\pi}$ must implement a $\exists$ transition to a new state, let us call this state $u_{x\exists}$. Since all π and $\exists$ transitions in M_{head} and M_1 up to M_n are loops we know that $u_{x\exists}$ is related to the same states as u_x in these specifications.

For the remainder of the proof, we do a case analysis on the mode of q:

- If $\mathsf{mode}(q) = \mathsf{Exst}$ then we know that (M_{ctrl}, q) has to implement an $\exists$-transition followed by a π-transition reaching a state q' that implements at least one state of form $\langle q\tau_m m a' d q'\rangle$. So if we extend $\mathsf{T}_{\langle M,w\rangle}$ at $\langle q, m, \tau\rangle$ with a new child $\langle q', m', \tau[\tau_m \mapsto a']\rangle$ then the new execution step will follow the semantics of the ATM T satisfying the inductive hypothesis both in part (1) and (2) — provided that $m' = m + 1$ if $d = \mathsf{r}$, and $m' = m - 1$ otherwise.
- If $\mathsf{mode}(q) = \mathsf{Univ}$ then we know that (M_{ctrl}, q) has to implement an $\exists$-transition followed by a π-transition reaching a state q' that implements one of the states $(M_{\text{ctrl}}, \langle q\tau_m m\rangle)$. The refinement relation with M_{head} and M_m ensures that this state is the only successor of q in M_{ctrl} that can be implemented, implying that $u_{x\pi}$ must implement all the transitions corresponding to the transition relation δ of T. Thus we can extend $\mathsf{T}_{\langle M,w\rangle}$ with new children $\langle q', m', \tau'[\tau_m \mapsto a']\rangle$ for all (q', m', τ') such that (M_{ctrl}, q') can be reached from $(M_{\text{ctrl}}, \langle q\tau_m m\rangle)$ in one step with a transition labeled (τ', m, a', d). Also $m' = m + 1$ if $d = \mathsf{r}$, and $m' = m - 1$ otherwise.
  It is not hard to see that all newly added successors maintain the inductive hypothesis, condition (1) and (2).

[7] In order to avoid repetitions we defer the argument for (2) by several paragraphs, as this discussion here would be essentially the same as for the inductive cases, due to the root being an existential or a universal state.

For all of these target states we now have to prove that the induction hypothesis holds. As all of the target states are reached by a transition in M_{ctrl} we know that there exists a state $u_y \in U_{\text{states}}$ such that $(u_y, q') \in Q_{\text{ctrl}}$. Because of the label on the transition we also know that $(u_y, p_l) \in Q_{\text{tape}}$ for $l = m + 1$ if $d = r$, and $l = m - 1$ if $d = l$. This is also ensured to be done in such a way that the tape cell specifications M_1 to M_n again match the content of the tape. We also know, because of all the transitions of type $(\_, \_, \_, \_)$ going from x_3 to x_1 in M_{exist}, that $(u_y, x_1) \in Q_{\text{exist}}$. This finishes the proof of the inductive step.

In this way we can recursively construct a pruned computation tree $\mathsf{T}_{\langle M,w\rangle}$. The constructed tree is finite, because we have argued that it follows the semantics of the ATM T, and T repeats no configuration along a single computation path. Moreover $\mathsf{T}_{\langle M,w\rangle}$ is accepting as it never is stuck in a rejecting (existential) state.

B Proof of Corollary 6

Since two of the six results were shown in [12], it suffices to show the four EXPTIME-completeness results, written in bold font in Table 2. Also in [12] we have argued that all of the problems considered here are in EXPTIME. The EXPTIME-completeness of CI for modal specifications follows directly from this fact and from Theorem 4 since each specification in $\mathcal{M}_w^T$ is a modal one, and since the sum of their sizes is polynomial in n and in the size of T

Since the decision problems CI, C, and TR are known to be in EXPTIME for mixed specifications, it suffices to give two reductions, one from CI for modal specifications to C for mixed specifications, and one from C for mixed specifications to TR for mixed specifications. Both reductions were already provided in [12], in the context of a PSPACE-hardness proof. We restate them here for the sake of completeness.

B.1 Reduction from CI for modal specifications to C for mixed specifications

It suffices to show how $k > 1$ modal specifications (M_i, s_i) can be conjoined into one mixed specification (M, c_k) with $|\,M\,|$ being polynomial in $\sum_i |\,M_i\,|$ such that (M, c_k) has an implementation iff all (M_i, s_i) have a common implementation.

Figure 9 illustrates the construction, which originates in [9], by showing a conjunction of states s_1, s_2, s_3 up to s_k. In order to conjoin two states s_1 and s_2, two new π-transitions are added from a fresh state c_2 to each of s_1 and s_2. One of the π-transitions is a may π-transition and the other is a must π-transition. Only two states can be conjoined directly in this way, but the process can be iterated as many times as needed, as seen in the figure, by adding a corresponding number of π-transitions to the newly conjoined systems. Observe that the resulting specification is properly mixed (not modal). Its size is linear in $\sum_i |\,M_i\,|$ and quadratic in k, which itself is $O(\sum_i |\,M_i\,|)$.

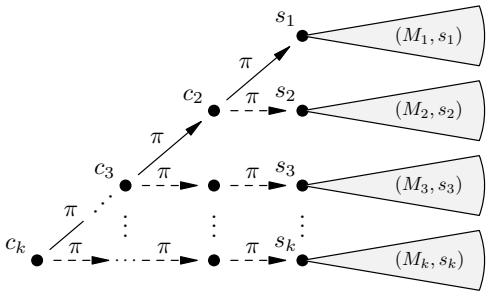


Fig. 9. Conjunction of k mixed (also modal) specifications into one mixed specification

If the specifications that are being conjoined have a common implementation, then the new specification will also have an implementation which is the same implementation prefixed with a sequence of $k-1$ π-transitions. Conversely if the new mixed specification has an implementation, then this implementation will contain at least a sequence of $k-1$ π-transitions, followed by an implementation that must individually satisfy all the systems that have been conjoined. □

B.2 Reduction from C for mixed specifications to TR for mixed specifications

It suffices to reduce C to TR for mixed specifications. Let (M, s) be a mixed specification over Σ. Consider a modal specification (N, t) over $\Sigma \cup \{\pi\}$ with $N = (\{t\}, \{\}, \{\})$, which only has a single state and no transitions. From (M, s) construct the mixed specification (M', s') over $\Sigma \cup \{\pi\}$ by prefixing s with a new state s' and a single transition $(s', \pi, s) \in R^{\Diamond}_{M'} \setminus R^{\Box}_{M'}$. Then (M', s') is a mixed specification that has (N, t) as an implementation, where $Q = \{(s', t)\}$ is the witnessing refinement relation. We show that (M, s) is consistent iff not $(N, t) \prec_{th} (M', s')$.

1° If (M, s) is consistent, then it has an implementation (L, l), from which we get an implementation (L', l') of (M', s') by creating a new state l' with a transition (l', π, l). But then (M', s') has an implementation that is not allowed by (N, t) and so $I(M', s') \not\subseteq I(N, t)$.

2° Conversely, if $I(M', s') \not\subseteq I(N, t)$ then there exists an implementation (L, l') of (M', s'), which is not an implementation of (N, t) – and so (L, l') has a transition (l', π, l). Moreover (L, l) refines (M, s) since (L, l') refines (M', s') and s is the unique successor of s' in M'. Thus (M, s) is consistent.

Remark: Observe that the first argument above would also work for refinement instead of thorough refinement. However we would not be able to get the second implication for refinement, due to its incompleteness. Also note that we have just shown EXPTIME-completeness not only for deciding whether a *mixed* specification thoroughly refines another *mixed* specification, but also for deciding whether a *mixed* specification thoroughly refines a *modal* specification. □